



مبانی برنامه‌نویسی

(۱۱-۱۳۰-۱۳۰۹)

جلسه‌ی هفدهم



دانشگاه شهید بهشتی

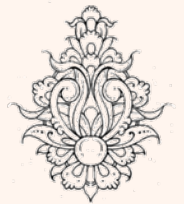
پاییز ۱۳۹۳

دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

فهرست مطالب

- مقدار برگشتی
- فراخوانی تابع
- فراخوانی تودرتو
- آرگومان و پارامتر
- ارسال با مقدار
- محدودی اعتبار متغیرها

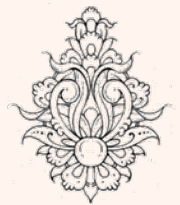


```
// function example
#include <iostream>
using namespace std;

int addition (int a, int b)
{
    int r;
    r=a+b;
    return (r);
}

int main ()
{
    int z;
    z = addition (5,3);
    cout << "The result is " << z;
    return 0;
}
```

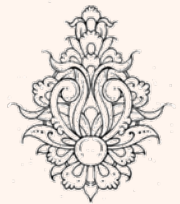
The result is 8



چگونگی کارکرد

- ابتدا تابع main اجرا می‌شود.
- دو مقدار ۵ و ۳ در متغیرهای a و b کپی شده و به تابع ارسال می‌گردند.
- مقدار مورد نظر در تابع محاسبه شده و مقدار ۸ باز می‌گردد.
- متغیرهای a و b تنها در محدوده‌ی تابع معتبر هستند.

```
int addition (int a, int b)
z = addition ( 5 , 3 );
```



```

int cube(int x)
{ // returns cube of x:
return x*x*x;
}

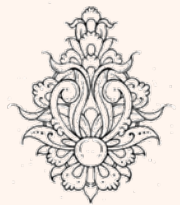
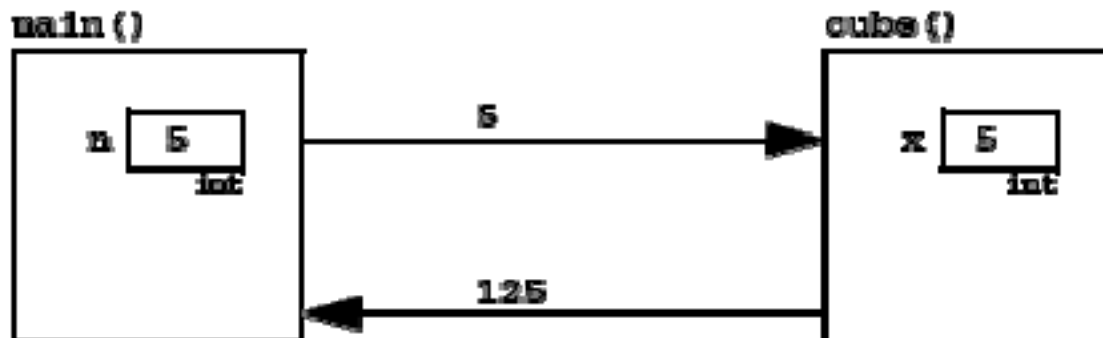
int main()
{ // tests the cube() function:
int n=1;
while (n != 0)
{
cout<<"Enter a number (0 for quit):";
cin >> n;
cout << "\tcube(" << n << ") = " << cube(n) << endl;
}
}

```

```

Enter a number (0 for quit):4
      cube(4) = 64
Enter a number (0 for quit):6
      cube(6) = 216
Enter a number (0 for quit):9
      cube(9) = 729
Enter a number (0 for quit):0
      cube(0) = 0

```

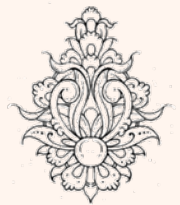


```
// function example
#include <iostream>
using namespace std;

int subtraction (int a, int b)
{
    int r;
    r=a-b;
    return (r);
}

int main ()
{
    int x=5, y=3, z;
    z = subtraction (7,2);
    cout << "The first result is " << z << '\n';
    cout << "The second result is " << subtraction (7,2) << '\n';
    cout << "The third result is " << subtraction (x,y) << '\n';
    z= 4 + subtraction (x,y);
    cout << "The fourth result is " << z << '\n';
    return 0;
}
```

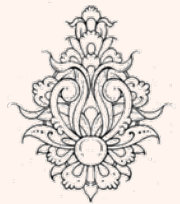
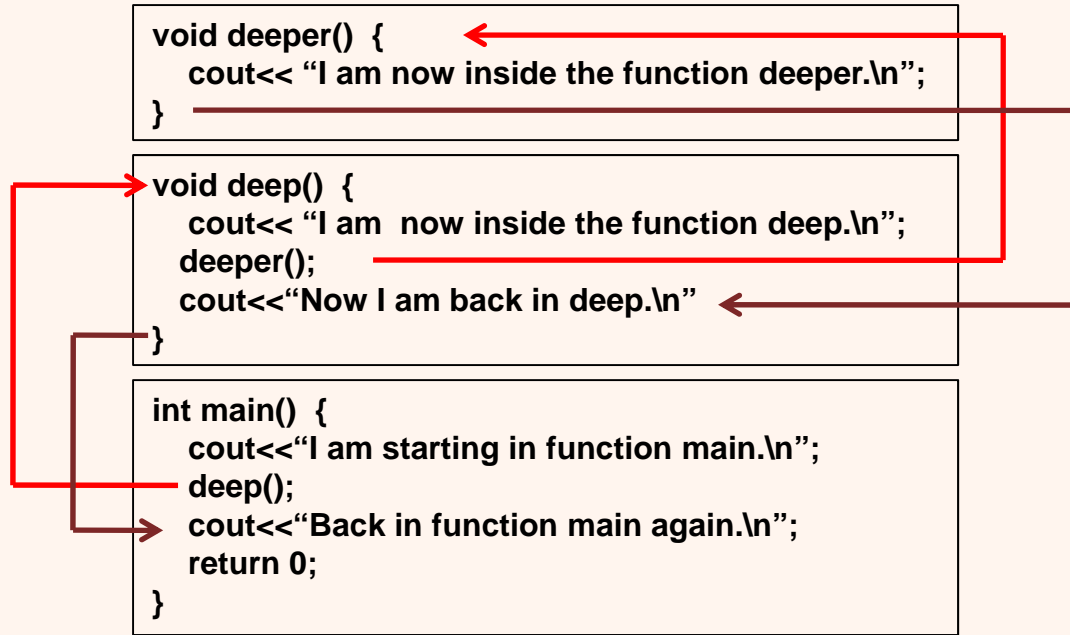
```
The first result is 5
The second result is 5
The third result is 2
The fourth result is 6
```



فرافوانی تودرتو

```
void deeper()
{
    cout << "I am now inside the function deeper.\n";
}
void deep()
{
    cout << "I am now inside the function deep.\n";
    deeper(); // Call function deeper
    cout << "Now I am back in deep.\n";
}
int main()
{
    cout << "I am starting in function main.\n";
    deep(); // Call function deep
    cout << "Back in function main again.\n";
    return 0;
}
```

```
I am starting in function main.
I am now inside the function deep.
I am now inside the function deeper.
Now I am back in deep.
Back in function main again.
```

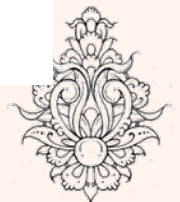


مثال

```
int max(int x,int y)
{ // returns larger of the two given integers:
if (x < y) return y;
else return x;
}
int main()
{ // tests the max() function:
int m,n;
do
{
cout<<"Enter 2 number (0 for quit):";
cin >> m >> n;
cout << "\tmax(" << m << "," << n << ") = " << max(m,n) << endl;
}
while (m != 0);
}
```

return برای تابع مانند break است برای ملقه یعنی هر زمان به return برسد از تابع بیرون می‌پرد. گر چه در بیشتر مواقع در پایان تابع است اما در هر جای دیگر نیز می‌تواند بیاید.

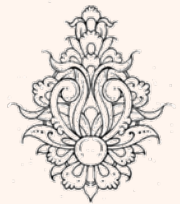
```
Enter 2 number (0 for quit):6 7
max(6,7) = 7
Enter 2 number (0 for quit):89 4
max(89,4) = 89
Enter 2 number (0 for quit):0 4
max(0,4) = 4
```



- قبل از اینکه کامپایلر تابعی را فراخوانی کند، نیاز دارد چیزهایی را جمع به تابع بداند:
 - نام تابع
 - تعداد و نوع آرگومان‌های ورودی
 - نوع آرگومان خروجی
- برای این منظور دو رویکرد وجود دارد:
 - اعلان تابع (**declaration** یا **function prototype**)

Function prototypes are also known as *function declarations*.

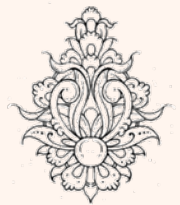
- قرار دادن بدنه‌ی تابع یا (**function definition**)



• راه‌های تعریف یک تابع:

- توابع قبل از تابع `main()` به طور کامل با بدنه مربوطه آورده شوند.
- راه دیگر، این گونه است که ابتدا تابع **اعلان** شود، سپس متن برنامه‌ی اصلی `main()` بیاید، پس از آن بدنه‌ی کامل تابع قرار بگیرد. (معمولا از این شیوه استفاده می‌شود)

اعلان تابع یا همان Prototype یک تابع، برای شناساندن تابع است که تنها شامل نام تابع و نوع آرگومان‌های ارسالی و نوع بازگشتی به همراه «;» است.



مثال

```
#include <iostream>
using namespace std;

int sub (int a,int b)
{
    int z;
    z=a-b;
    return z;
}

int main ()
{
    int x=10, y=3;
    cout << x<<"-"<<y<<"="<<sub (x,y)<< endl;
    return 0;
}
```

```
1
#include <iostream>
using namespace std;
int sub (int ,int );
int main ()
{
    int x=10, y=3;
    cout << x<<"-"<<y<<"="<<sub (x,y)<< endl;
    return 0;
}
int sub (int a,int b)
{
    int z;
    z=a-b;
    return z;
}
```

آرگومان‌ها و پارامترها

- مقادیری که به تابع ارسال می‌شوند، **آرگومان** و متغیرهایی که مقادیر مذکور را دریافت می‌نمایند **پارامتر** نامیده می‌شوند.

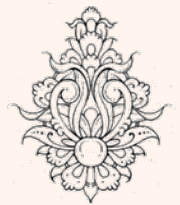
```
#include <iostream>
using namespace std;
void displayValue(int);
int main()
{
    cout << "I am passing 5 to displayValue.\n";
    displayValue(5);
    cout << "Now I am back in main.\n";
    return 0;
}
```

```
I am passing 5 to displayValue.
The value is 5
Now I am back in main.
```

آرگومان ارسالی

```
void displayValue(int num)
{
    cout << "The value is " << num << endl;
}
```

پارامتر تابع



```
void displayValue(int);
```



```
void displayValue(int num);
```



function prototype

هر دو نمونه برای تعریف **prototype** صحیح است هر چند تنها مشخص کردن نوع کافیست و در مورد دوم کامپایلر **num** را در نظر نمی‌گیرد.

```
displayValue(5);
```

```
// function call
```

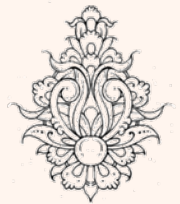
```
void displayValue(int num)
```

```
// function header
```

```
{
```

```
    cout << "The value is " << num << endl;
```

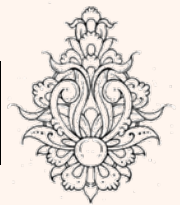
```
}
```



- تابعی بنویسید که دو عدد a و p را دریافت و a را به توان p برساند.

```
#include <iostream>
using namespace std;
int power(int ,int );
int main()
{ int a,p;
cout<<"enter two number:";
cin>> a >> p;
cout <<"a^p=" <<power(a,p)<<endl;
}
int power(int k,int l)
{
int i,n=1;
for(i=1;i<=l;i++)
n=n*k;
return n;
}
```

```
enter two number:2 5
a^p=32
```

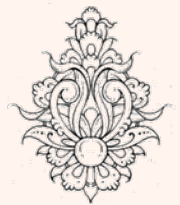


- تابعی بنویسید که عددی را گرفته فاکتوریل عدد را محاسبه نماید.

```
long fact(int n)
{ // returns n! = n*(n-1)*(n-2)*...*(2)*(1)
if (n < 0) return 0;
int f = 1;
while (n > 1)
    f *= n--;
return f;
}
```

0 1 1 2 6 24 120

```
int main()
{ // tests the factorial() function:
for (int i=-1; i < 6; i++)
    cout << " " << fact(i);
cout << endl;
}
```



- تابعی بنویسید که فرمول جایگشت را پیاده‌سازی نماید:

$$P(n, k) = \frac{n!}{(n-k)!}$$

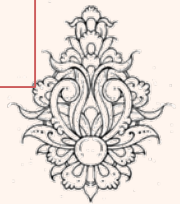
$$P(5, 2) = \frac{5!}{(5-2)!} = \frac{5!}{3!} = \frac{120}{6} = 20$$

```
long fact(int n)
{ // returns n! = n*(n-1)*(n-2)*...*(2) (1)
  if (n < 0) return 0;
  int f = 1;
  while (n > 1)
    f *= n--;
  return f;
}
```

```
long perm(int n, int k)
{ // returns P(n,k), the number of permutations of k from n:
  if (n < 0 || k < 0 || k > n) return 0;
  return fact(n)/fact(n-k);
}
```

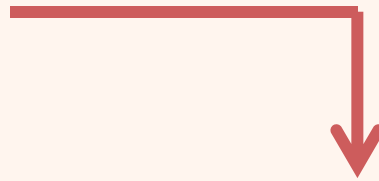
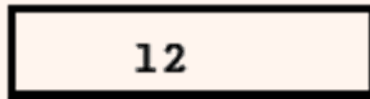
```
0 0
0 1 0
0 1 1 0
0 1 2 2 0
0 1 3 6 6 0
0 1 4 12 24 24 0
0 1 5 20 60 120 120 0
0 1 6 30 120 360 720 720 0
0 1 7 42 210 840 2520 5040 5040 0
```

```
int main()
{ // tests the perm() function:
  for (int i = -1; i < 8; i++)
  {
    for (int j = -1; j <= i+1; j++)
      cout << " " << perm(i, j);
    cout << endl;
  }
}
```

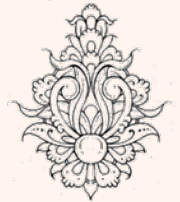
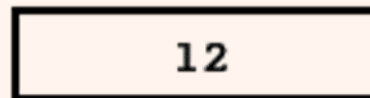


- هنگامی که آرگومانی به تابع از طریق مقدار ارسال می‌گردد، در اصل یک کپی از آرگومان ایجاد می‌گردد و فرآیندها به روی کپی صورت می‌پذیرد.
- تغییرات به روی کپی یا همان پارامترها، اثری به روی مقدار اصلی نخواهد داشت.

Original Argument
(in its memory location)



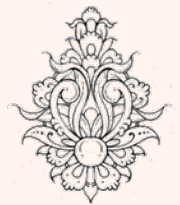
Function Parameter
(in its memory location)



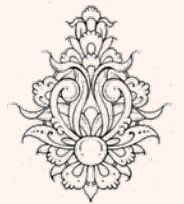
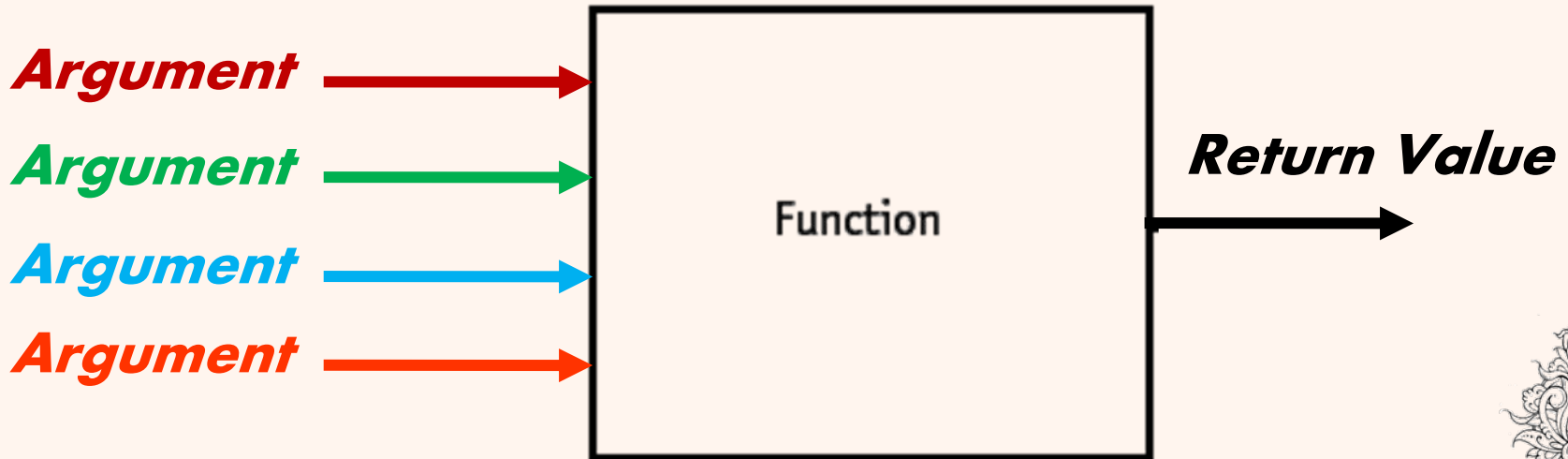
مثال

```
void func1(double, int); // Function prototype
int main()
{
int x = 0;
double y = 1.5;
cout << x << " " << y << endl;
func1(y, x);
cout << x << " " << y << endl;
return 0;
}
void func1(double a, int b)
{
cout << a << " " << b << endl;
a = 0.0;
b = 10;
cout << a << " " << b << endl;
}
```

```
0 1.5
1.5 0
0 10
0 1.5
```



- یک تابع را سیستمی در نظر بگیرید که دارای چندین کانال ورودی و تنها یک کانال فروجی است:



تابع تنها یک مقدار فروجی دارد

- مقدار بازگشتی را می‌توان به صورت معمولی در عبارات ریاضی استفاده نمود:

```
result = square (number);
```

20

```
int square (int number)
{
```

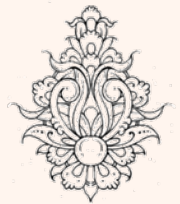
```
    return number * number;
```

```
}
```

400

```
if (square(number) > 100)
    cout << "big square\n";
```

```
sum = 1000 + square(number);
```



مقدار بازگشتی Boolean

Returning a Boolean Value

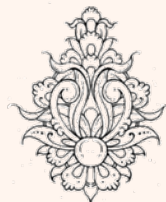
- مقدار بازگشتی تابع می‌تواند از جنس درست یا نادرست باشد که معمولاً برای بررسی حالات مختلف در برنامه از آن استفاده می‌شود.

```
// Function prototype
bool isEven(int);
```

```
int main()
{
    int val;
    cout << "Enter an integer and I will tell you ";
    cout << "if it is even or odd: ";
    cin >> val;
    if (isEven(val))
        cout << val << " is even.\n";
    else
        cout << val << " is odd.\n";
    return 0;
}
```

```
Enter an integer and I will tell you if it is even or odd: 67
67 is odd.
```

```
bool isEven(int number)
{
    if (number % 2)
        return false; // The number is odd if there's a remainder.
    else
        return true; // Otherwise, the number is even.
}
```



سی تولید منو

```
void displayMenu();
int getChoice();
void computeFees(char, double, int);
const double ADULT_RATE = 40.00, SENIOR_RATE = 30.00,
CHILD_RATE = 20.00;
int main(){
    int choice, // Holds the user's menu choice
    months; // Number of months being paid
    do{
        displayMenu();
        choice = getChoice(); // Assign choice
        // to it by the getChoice function.
        if (choice != 4){
            cout << "For how many months? ";
            cin >> months;
            switch (choice){
                case 1: computeFees('A', ADULT_RATE, months);
                    break;
                case 2: computeFees('C', CHILD_RATE, months);
                    break;
                case 3: computeFees('S', SENIOR_RATE, months);
            }
        }
    } while (choice != 4);
    return 0;
}
```

```
void displayMenu()
{
    cout << "\n\tHealth Club Membership Menu\n\n";
    cout << "1. Standard Adult Membership\n";
    cout << "2. Child Membership\n";
    cout << "3. Senior Citizen Membership\n";
    cout << "4. Quit the Program\n\n";
}
```

```
int getChoice()
{
    int choice;
    cin >> choice;
    while (choice < 1 || choice > 4)
        { cout << "The only valid choices are 1-4. Please re-enter. ";
          cin >> choice;
        }
    return choice;
}
```

```
void computeFees(char memberType, double rate, int months){
    cout << endl
        << "Membership Type : " << memberType << endl
        << "Monthly rate $" << rate << endl
        << "Number of months: " << months << endl
        << "Total charges : $" << (rate * months)
        << endl << endl;
}
```

```
Health Club Membership Menu
1. Standard Adult Membership
2. Child Membership
3. Senior Citizen Membership
4. Quit the Program

1
For how many months? 2

Membership Type : A
Monthly rate $40
Number of months: 2
Total charges : $80
```

هنگامی که کاربر انتخابی خاص از منو داشته باشد تابع متناظر اجرا می شود